REMARKS

Applicant appreciates the time taken by the Examiner to review Applicant's present application. This application has been carefully reviewed in light of the Official Action mailed March 4, 2005. Applicant respectfully requests reconsideration and favorable action in this case.

Rejections under 35 U.S.C. § 103

Claims 1-45 have been cancelled and new claims 46-81 have been added. Claim 46 recites a software program on a client computer that is executable to "receive a file at the client computer directly from a database; store the file as a cached file in the local cache; notify the operating system to open the cached file using a locally running application associated with the file type for the cached file; determine if the cached file at the client computer has been modified by the locally running application based on a notification from a file management system of an operating system; and if the cached file has been modified, save the cached file from the cache directly to the database." These features of the present invention allow a user using a client computer to request a file (e.g., a ".doc" file) from a document management database, cache the file at the local computer and open the file using a locally running program (e.g., Microsoft Word). When the user modifies the file using the locally running application of choice, the software program of the present invention detects the modification based on a notification from the file management system of the operating system. The software program of the present invention can then save the cached file to the database thereby saving the modifications to the database. This allows, for example, a user working on .doc documents to seamlessly modify documents in an underlying database. Moreover, the software program of the present invention is agnostic as to the file type. Thus, the software program can retrieve and cache files of any file type and allow the user to work on the file type with any locally running application that the user wishes to use to modify the particular file.

Prior systems of document management or other knowledge management would require a second program (e.g., a "synchronization program") on the client machine to synchronize any modifications the user makes on his/her client machine with the database. These synchronization programs are unattractive because they typically run with only one

software tool or require significant amounts of coding to run with multiple tools. Moreover, these systems often required a user to save the document using the synchronization tool, requiring an extra step in the save process. Other prior art systems map databases to virtual local drives (e.g., an "E" or "F" drive). However, these systems often require additional programming at an operating system level.

Applicant respectfully submits that Pace focuses on distributing particular pieces of content (including database content and files) as part of a sub-application that can run on the user computer to view and/or modify the data. For example, Pace deals with sending content such as reference data or entity data (e.g., database data) with content the can be used to present the reference data or entity data (e.g., dynamic content, java bean, session bean, entity bean). *See*, Pace ¶¶ 349-351. Assets for a package are retrieved from a source tier to a deployment tier that includes an asset cache (the deployment tier includes the CDS/ADS). Assets are deployed from the deployment tier to the appropriate target. *See*, Pace, FIGURE 2A. Thus, the cache of the CDS/ADS cited by the Examiner is a cache at the deployment tier not the target tier utilized by users.

As part of determining the version of an asset to deploy, an export asset adapter (illustrated as 1600 in FIGURE 2) employs a version asset adapter method, which is described in more detail in conjunction with FIGURES 16A and mentioned in ¶790 (the paragraph cited by the Examiner). The asset adapter of the source tier, as described at ¶¶ 582-589 of Pace, calls a version asset adapter that returns the time stamp of assets in the source tier (the EIS) to determine the newest version of the data at the source tier. In other words, the version asset adapter "looks for the most recent version" for the asset in the EIS and returns the timestamp for that asset to ensure that the current version of the asset is deployed. *See*, Pace ¶583. Thus, the versioning system cited by the Examiner is drawn to ensuring that the most recent version of an asset is deployed to the target tier for viewing by a user, not determining if the user has modified that asset in the cache at the user computer.

When database data is deployed to a target, the client deployment asset adapter creates the table at the client and imports the data into the table. ¶¶643-644. A client database management system which can be deployed to the client in the BE layer of a package (*see*, ¶¶333 and 433), notifies the CDA when a database record has changed, passing arguments to indicate the asset type. ¶700. A synchronization asset adapter (SAA) retrieves the table updates, insertions or deletions and builds a "synchronization asset". ¶703. The

synchronization asset is then passed to the deployment tier (the CDS/ADS), which determines the appropriate source node for the asset. The CDS/ADS then transfers the synchronization asset to the appropriate source node. ¶¶ 707-710.

Thus, in Pace, the database environment of the source is essentially replicated at the client to allow a user to view particular database records. If the records are changed, the database management system of the client determines that a change has been made and calls the appropriate adapter to send the changes to the deployment tier. The deployment tier then sends the changes to the source tier. Because the sections of Pace cited by the Examiner are drawn to the modification of database table information utilizing a client DBMS, there is no teaching or suggestion that a software program should receive a notification that a cached file has changed from a file management system of the operating system.

The Examiner cites Kishi as showing a notification from a file management system of an operating system. More particularly, the Examiner states that "it would have been obvious to combine the teachings of the cited references because Kishi's teaching would have allowed Pace's to reconcile a server and client database of a file system via an automatic storage manager sending a Start message based on a timer function when the distributed storage manager server can reconcile (col. 5, lines 18-35)."

Kishi deals with a system for backing up data from a network to tapes. The VTS system provides a hard drive buffer (the DASD). This allows data to be written by backup applications to the relatively fast DASD and then from DASD to the slower tapes. Kishi is drawn to a system for handling the writing of data from the DASD to the physical backup tapes. The START message sent by the system of Kishi is not a notification that a particular file has changed, but is simply a message to start reconciliation between the DASD and the physical tapes because a certain amount of time has passed.

It is unclear how the references can be combined in the manner suggested by the Examiner because Pace deals with the distribution and management of content to target computers, while Kishi deals with the reconciliation of a DASD and physical tape media in an enterprise backup system. However, even if combined, the DMBS of Pace would still make the determination that a particular record has been modified at the client computer and that synchronization should be performed. The START type message of Kishi would simply cause reconciliation at an arbitrarily specified time period regardless of whether records have changed.

Kishi does discuss that a VTS function call can be made to the automatic storage manager administrator supplied function to notify the automatic storage manager administrator of a file closing. However, this appears to be a file closing on the DASD (i.e., the hard drive system of the overall backup system) and not a file being modified at a local user machine. It is unclear how this system would function to determine that a database table of Pace had been updated as the database tables are monitored by the DBMS of Pace. Finally, Pace already provides a sufficient mechanism for determining that a modification has been to the database data at the client using the client DMBS, not a file management system. There would therefore be no motivation to modify Pace in the a manner of the claimed invention.

As the references do not teach or suggest "determine if the cached file at the client computer has been modified by a user using the locally running application based on a notification from a file management system of an operating system." Applicant respectfully requested allowance of Claim 46 and the respective dependent Claims. For similar reasons, Applicant submits that Claims 59-81 are also distinguishable from the cited art.

Applicant has now made an earnest attempt to place this case in condition for allowance. Other than as explicitly set forth above, this reply does not include an acquiescence to statements, assertions, assumptions, conclusions, or any combination thereof in the Office Action. For the foregoing reasons and for other reasons clearly apparent, Applicant respectfully requests full allowance of the pending Claims. The Examiner is invited to telephone the undersigned at the number listed below for prompt action in the event any issues remain.

The Director of the U.S. Patent and Trademark Office is hereby authorized to charge any fees or credit any overpayments to Deposit Account No. 50-3183 of Sprinkle IP Law Group.

Respectfully submitted,

Sprinkle IP Law Group
Attorneys for Applicant

John L. Adair
Reg. No. 48,828

Date: June 06, 2005
1301 W. 25th Street, Suite 408
Austin, TX 78705
Tel. (512) 637-9220
Fax. (512) 371-9088